Genetic Algorithms Based on Primal-Dual Chromosomes for Royal Road Functions

SHENGXIANG YANG
Department of Mathematics and Computer Science
University of Leicester
University Road, Leicester LE1 7RH
UNITED KINGDOM
s.yang@mcs.le.ac.uk http://www.mcs.le.ac.uk/~syang

Abstract: Genetic algorithms (GAs) have been broadly studied by a huge amount of researchers and there are many variations developed based on Holland's simple genetic algorithm (SGA). Inspired by the idea of diploid genotype and dominance mechanisms that broadly exists in nature, we propose a primal-dual genetic algorithm (PDGA). PDGA operates on a pair of chromosomes that are primal-dual to each other in the sense of Hamming distance in genotype. We compare the performance of PDGA over SGA based on the Royal Road functions, which are specially designed for testing GA's performance. The experiment results show that PDGA outperforms SGA on the Royal Road functions for different performance measures.

Key-Words: Genetic algorithm, primal-dual chromosomes, schema, diploid, dominance, royal road functions.

1 Introduction

Genetic algorithms (GAs) were first proposed by Holland [6], usually called Holland's canonical or simple genetic algorithm (SGA), and then developed by Holland and his colleagues and students in the 1960s and the 1970s. Thereafter, there have been many variations and extensions developed based on Holland's SGA [5, 7, 9], including macro-structure and micro-structure variations.

GA's macro-structure variations include hybrid GAs, master-slave GAs, and parallel GAs. In GA's microstructure aspect, variations appear on every aspect of GA's components, including chromosome representation schemes, selection strategies (e.g., the elitist model, the ranking model, and several scaling strategies), mating policies (e.g., sharing function, niche, and seduction techniques), crossover operators (e.g., one-point, two-point and uniform crossover operators), and mutation operators [5, 7, 9].

Though most GAs studied so far are based on the haploid or single-stranded chromosome, which is the simplest genotype found in nature, GA's researchers have also studied diploid genotypes (pairs of chromosomes) and dominance mechanisms for a long history [1, 2, 4, 10]. In the diploid form a genotype carries one pair of chromosomes (called homologous chromosomes). Each position or locus in the genome has two or multiple allele values. A dominance mechanism for determining which allele value for a gene will be expressed is required to adjudicate when the allele values do not agree [5].

An issue of great concern about GA is the balance between exploration (the investigation for new, useful adaptations in the search space) and exploitation (the use and propagation of these adaptations). An efficient algorithm is one that uses both techniques. In this paper, inspired by the idea of diploid genotype and dominance mechanisms that broadly exist in nature, we present a primal-dual genetic algorithm (PDGA) based on the concept of primal-dual chromosomes. PDGA operates on a pair of chromosomes that are primal- dual to each other in the sense of Hamming distance in genotype. Through the primal-dual or Hamming distance mapping between a pair of chromosomes, GA's performance of exploration in the search space is improved and thus its total searching efficiency is improved.

In our present study, we provide a comparison of the performance of PDGA over SGA based on the Royal Road functions [3, 8, 9]. These functions are specially designed for testing the performance of GAs and thus can act as good test problems. The experiment results show that PDGA outperforms SGA for different performance measures.

2 Primal-Dual Genetic Algorithms

2.1 Definition of Primal-Dual Chromosomes

Here we consider binary bit string representation of genotype and define a pair of chromosomes to be *primal-dual* to each other if their Hamming distance (the number of locations at which corresponding bits

differ) is the maximum (equal to their length). That is, given a chromosome $x = (x_1,...,x_L) \in I = \{0,1\}^L$ of fixed length L, its dual chromosome is defined as $\bar{x} = (\bar{x}_1,...,\bar{x}_L) \in I$ where $\bar{x}_i = 1 - x_i$ $(i = 1..L)$.

Given above definition, we can say that x is mapped to $\bar{x}$ by primal-dual mapping or Hamming distance mapping, vice versa. And the dominance mechanism works on the primal-dual pair by taking whichever has higher fitness as the dominant chromosome, i.e., the dominance is phenotype-based.

2.2 Framework of PDGA

PDGA is quite simple compared to other genetic algorithm variations. It can be described in a step-by-step format as follows:

Step 1. Set parameters such as population size N, crossover probability P_c, mutation probability P_m, and maximum allowable generation number $t_{\max}$. Initialise a random population. Set counter $t = 0$.
Step 2. In the population, evaluate each chromosome x's fitness and its dual chromosome $\bar{x}$'s fitness. If $\bar{x}$ is better, replace x with $\bar{x}$.
Step 3. If $t > t_{\max}$ or some other termination condition is satisfied, stop.
Step 4. Select chromosomes with some policy from the population to generate the mating pool.
Step 5. Perform crossover on chromosome pairs randomly chosen from the mating pool.
Step 6. Perform mutation on each chromosome.
Step 7. In the new population, evaluate each chromosome x's fitness and its dual chromosome $\bar{x}$'s fitness. If $\bar{x}$ is better, replace x with $\bar{x}$.
Step 8. Perform elitist selection. This step is optional.
Step 9. Set $t = t + 1$, and go to Step 3.

Here we can see that PDGA differs from SGA only in Step 2 and Step 7, i.e., in the evaluation of chromosomes in the population. The genetic operations including selection, crossover and mutation are all the same for both PDGA and SGA.

2.3 Haploid or Diploid?

From the above description of PDGA, it seems that it is diploidy-based since it works on a pair of primal-dual chromosomes. However, further thinking will make it a haploid-like genetic algorithm since in fact we needn't explicitly keep track of each chromosome's dual chromosome. That is, a dual chromosome can be looked as the shadow of its primal chromosome and only shows its body (through the primal-dual mapping) when the primal chromosome is evaluated. Of course, if the dual chromosome proves to be better, it will embody itself and throw the primal chromosome into its shadow. In this sense, we can call PDGA a *pseudo-diploid* or *implicitly diploid* genetic algorithm.

Here we must also note that PDGA is different from those genetic algorithms [1, 2, 4] that are based on diploidy and dominance in the following three aspects: First, in those GAs dominance is gene oriented and thus needs special dominant scheme while in PDGA dominance is phenotype oriented and thus needs no special dominant scheme. Second, in those GAs the chromosomes undergoing dominance operation are chosen randomly while in PDGA dominance operates on the primal-dual pair of chromosomes. Finally, in those GAs genetic operations perform on the pair of chromosomes while in PDGA only the winner or dominant of the primal-dual pair goes through genetic operations.

3 Building Blocks and Royal Road

3.1 Building Blocks and Schema Theorem

The building block hypothesis and schema theorem of Holland are the theoretical foundations of GA [6]. Holland first proposed the notation of *schema* to describe a subset of all binary vectors of fixed length that have similarities at certain positions. A schema is typically specified by a vector over the alphabet {0, 1, ∗}, where the "∗" denotes a "wildcard" matching both 0 and 1. Given a schema S, its *order* $o(S)$ is simply the number of fixed positions within S and its *defining length* $l(S)$ is the maximum distance between fixed positions within S. For example, given $S = 01{*}{*}1$, $o(S) = 3$ and $l(S) = 4$.

The building block hypothesis states that the GA works best when short, low-order, high-fit schemas that contain the optimum or desired near-optimum recombine to form even more highly fit higher-order schemas. The schema theorem states that short, low-order, better than average schemas (also called building blocks) receive an exponentially increasing number of trials in the subsequent generations.

3.2 Royal Road Functions

The schema theorem does not state how crossover, the major source of the search power of GAs, works to recombine highly fit schemas. The building block hypothesis states that crossover combines short, observed high-fit schemas into increasingly fit candidate solutions, but doesn't give any detailed description of how this combination occurs.

To investigate schema processing and recombination in detail, Mitchell, Forrest and

Holland designed a class of fitness landscapes, called Royal Road functions, to capture the essence of building blocks in an idealized form [3, 8, 10]. For example, Fig.1 and Fig.2 show the Royal Road functions R_1 and R_2 respectively.

```
S1  = 11111111************************************************************
S2  = ********11111111********************************************************
S3  = ****************11111111****************************************************
S4  = ************************11111111************************************************
S5  = ********************************11111111********************************************
S6  = ****************************************11111111****************************************
S7  = ************************************************11111111************************
S8  = ********************************************************11111111
Sopt = 11111111111111111111111111111111111111111111111111111111111111111111
```

Fig.1 Royal Road Function R_1.

```
S1  = 11111111************************************************************
S2  = ********11111111********************************************************
S3  = ****************11111111****************************************************
S4  = ************************11111111************************************************
S5  = ********************************11111111********************************************
S6  = ****************************************11111111****************************************
S7  = ************************************************11111111************************
S8  = ********************************************************11111111
S9  = 1111111111111111************************************************************
S10 = ****************1111111111111111********************************************
S11 = ********************************1111111111111111********************************
S12 = ************************************************1111111111111111
S13 = 11111111111111111111111111111111************************************************
S14 = ********************************11111111111111111111111111111111
Sopt = 11111111111111111111111111111111111111111111111111111111111111111111
```

Fig.2 Royal Road Function R_2.

Royal Road functions R_1 and R_2 contain tailor-made building blocks and thus are good test problems to investigate GA's performance with respect to schema processing and recombination. They are defined using a list of schemas s_i. Each s_i is given a coefficient c_i equal to its order (i.e., $c_i = o(s_i)$). From Fig. 1 and Fig. 2, it can be seen that for R_1, $c_i = 8$ for all s_i (i = 1..8) while for R_2, $c_i = 8$ for s_i (i = 1..8), $c_i = 16$ for s_i (i = 9..12), and $c_i = 32$ for s_i (i = 13, 14). The fitness of a bit string x for both $R_1(x)$ and $R_2(x)$ is computed by summing the coefficients c_i corresponding to each of the given schema s_i of which x is an instance, shown as follows:

$$R_1(x) = \sum_{i=1}^{i=8} c_i \delta_i(x),\ R_2(x) = \sum_{i=1}^{i=14} c_i \delta_i(x)$$

where $\delta_i(x)$ = {1, *if* $x \in s_i$; 0, *otherwise*}. For example if x is an instance of exactly two of the order-8 schemas of R_1, $R_1(x) = 16$. Similarly, $R_1(s_{opt}) = R_1(11..1) = 64$, $R_2(s_{opt}) = R_2(11..1) = 192$.

For Royal Road functions, with PDGA we can evaluate the primal-dual chromosome pair at the same time with only a few extra instructions than when we evaluate one single chromosome with SGA. For example, the pseudo-code of procedure $R_1(x)$ that evaluates a bit string $x = (x_1, x_2, .., x_L)$ of fixed length L is shown as follows.

```
   Procedure R1(x):
1  onesCount :=0, dualFitness :=0, primFitness := 0;
2  for i := 1 to L do
3    if x_i =1 then onesCount := onesCount +1; endif;
4    if (i MOD 8) = 0 then
5      if onesCount = 8 then
6        primFitness := primFitness + 8;
7      else if onesCount = 0 then
8        dualFitness := dualFitness + 8;
9      endif;
10     onesCount := 0;   /* clear the counter */
11   endif;
12 endfor;
13 if dualFitness > primFitness then  /* replace */
14   primFitness := dualFitness;
15   for i := 0 to L do  x_i  :=  1 -  x_i ; endfor;
16 endif;
```

From above pseudo-code it can be seen that only lines 7, 8, 13 to 16 are extra for PDGA than SGA. Here we gain an almost free lunch in the sense of function evaluation number. And for this reason we count the evaluations of the primal-dual pair as one evaluation in our following experiments.

4 Computer Experiment Study

In all our experiments, we use sigma truncation scaling and elitism selection schemes, one-point crossover and bit mutation for both PDGA and SGA.

4.1 Performance Measures

Here the performance measures have twofold functions: one is to test the absolute performance of PDGA itself as a genetic algorithm; the other is to compare the relative performance of PDGA over SGA. With this consideration, we use the following performance measures: function evaluations to optimum, percentage of achieving optimum over a number of runs against generations, percentage of optimal members in the population for a run against generations.

4.2 Design of Experiments

In our primary experiments, we consider the effects of various parameter settings of N (128,

1024), P_c (0.6, 0.7, 0.8) and P_m (0.005, 0.01, 0.02) on the performance of PDGA over SGA using the performance measure of function evaluations to optimum. We carried out 200 runs of SGA and PDGA on R_1 and R_2 respectively using the same 200 random seeds for each parameter setting and recorded the function evaluations required to obtain their optimal solutions. Here only those chromosomes changed by crossover and mutation operations were evaluated. The statistic results with respect to mean and best (or least) evaluations to optimum over 200 runs are given in Table 1 and Table 2. Fig.3 shows the result more intuitively.

From Table 1, Table 2 and Fig.3 it can be seen that PDGA outperforms SGA on R_1 and R_2 with respect to both mean and best evaluations to optimum for most parameter settings. The best improvement of PDGA over SGA with respect to mean evaluations to optimum reaches about 25% with parameter setting 9 on R_1 and about 28% with parameter setting 18 on R_2.

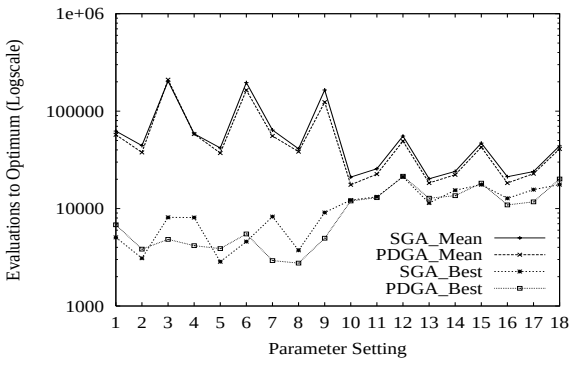
From Table 1, Table 2 and Fig.3, it can also be seen that the parameter setting does matter. Generally speaking, the effect of parameter setting is similar for both R_1 and R_2. The setting of P_c obviously doesn't have much effect and shows little difference between 0.6, 0.7 and 0.8. However the setting of N and P_m has

Table 1 Statistic results over 200 runs of PDGA vs. SGA on R_1 with respect to evaluations to optimum.

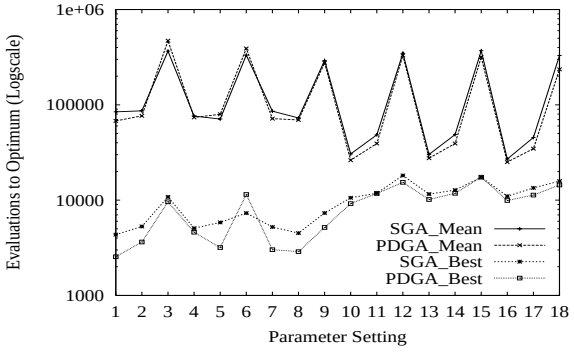
Parameter Setting				SGA		PDGA	
No.	N	P_c	P_m	Mean	Best	Mean	Best
1	128	0.6	0.005	62439	5063	57273	6818
2	128	0.6	0.01	44328	3094	37707	3834
3	128	0.6	0.02	203417	8133	210281	4834
4	128	0.7	0.005	58598	8091	58301	4164
5	128	0.7	0.01	41927	2846	37167	3883
6	128	0.7	0.02	196284	4578	164347	5487
7	128	0.8	0.005	63944	8272	55604	2926
8	128	0.8	0.01	41134	3733	38391	2754
9	128	0.8	0.02	164875	9105	123666	4961
10	1024	0.6	0.005	21054	12233	17172	11968
11	1024	0.6	0.01	25641	13212	22764	12946
12	1024	0.6	0.02	55531	21173	47431	21463
13	1024	0.7	0.005	20234	11414	17574	12772
14	1024	0.7	0.01	24072	15424	23343	13555
15	1024	0.7	0.02	47036	17593	43530	18248
16	1024	0.8	0.005	21239	12735	18367	10900
17	1024	0.8	0.01	23992	15659	23228	11724
18	1024	0.8	0.02	44040	17624	45797	20126

Table 2 Statistic results over 200 runs of PDGA vs. SGA on R_2 with respect to evaluations to optimum.

Parameter Setting				SGA		PDGA	
No.	N	P_c	P_m	Mean	Best	Mean	Best
1	128	0.6	0.005	84491	4343	67761	2544
2	128	0.6	0.01	86583	5291	76711	3635
3	128	0.6	0.02	365759	10805	471597	9607
4	128	0.7	0.005	76212	5072	73837	4602
5	128	0.7	0.01	70939	5831	79641	3182
6	128	0.7	0.02	332978	7305	389978	11456
7	128	0.8	0.005	85816	5223	71700	3018
8	128	0.8	0.01	72881	4505	69481	2878
9	128	0.8	0.02	292604	7322	276523	5168
10	1024	0.6	0.005	30539	10581	26236	9199
11	1024	0.6	0.01	48487	11825	39295	11733
12	1024	0.6	0.02	348073	18128	329780	15376
13	1024	0.7	0.005	30178	11577	27566	10157
14	1024	0.7	0.01	48701	12754	39277	11577
15	1024	0.7	0.02	369536	17373	316985	17414
16	1024	0.8	0.005	26965	10990	25085	9918
17	1024	0.8	0.01	45463	13438	34698	11300
18	1024	0.8	0.02	329686	15823	235858	14419



(a)



(b)

Fig.3 Effects of parameter setting on the performance of mean and best function evaluations to optimum of PDGA vs. SGA on (a) R_1 and (b) R_2.

great effects and correlation exists between N and P_m. Generally speaking, greater population size 1024 is much better than smaller population size 128 (except on R_2 when $P_m = 0.02$). When $N = 128$, setting P_m to 0.01 is much better than to 0.005 and 0.02 while when $N = 1024$, $P_m = 0.005$ is the best.

For our further experiments, we fix the parameters as follows: $N = 128$, $P_c = 0.7$, $P_m = 0.01$. We set N to 128 instead of 1024 for the sake of better analyses and comparisons between PDGA and SGA because when $N = 1024$ they both converge quite fast with respect to generations to optimum (about 40 to 60 generations on the average).

4.3 Experiments on Achieving Optimum

To compare the dynamic performance of achieving optimum of PDGA over SGA, we carried out 1000 runs of PDGA and SGA on R_1 and R_2 under the same 1000 random seeds and with $t_{\max}$ set to 1000 for each run. For each run the information about whether optimum is achieved by current generation is reported every 25 generations. The statistic results with respect to percentage of achieving optimum and average percentage of optimal individuals in the population against generations over 1000 runs are shown in Fig.4 and Fig.5 respectively.

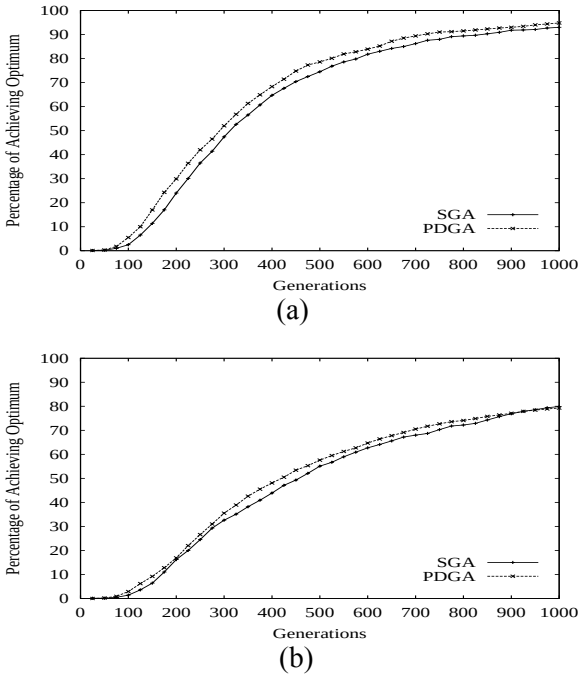


Fig.4 Comparisons of PDGA vs. SGA with respect to percentage of achieving optimum over 1000 runs against generations on (a) R_1 and (b) R_2.

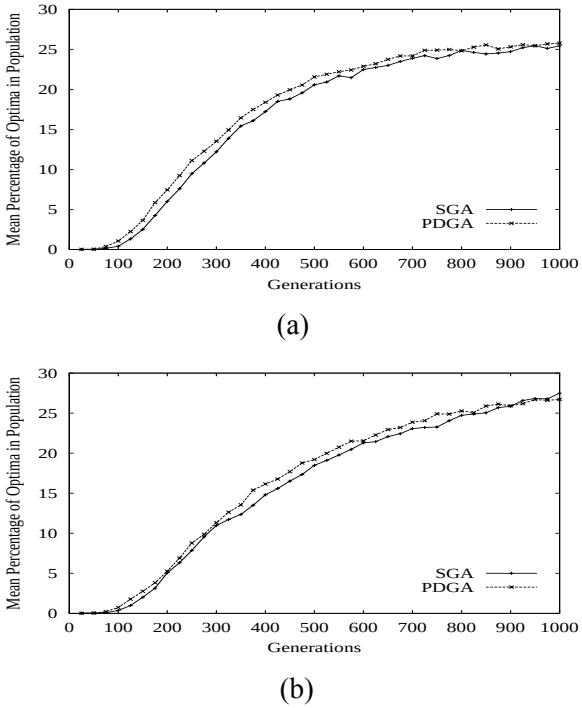


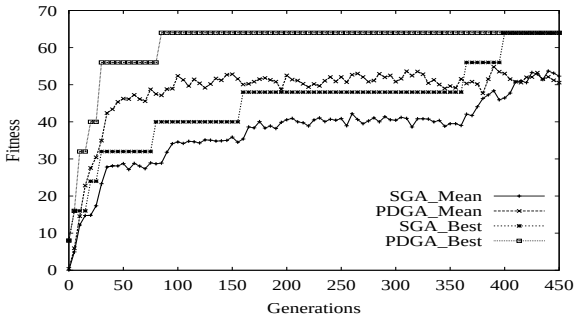
Fig.5 Comparisons of PDGA vs. SGA with respect to mean percentage of optimal individuals in the population against generations on (a) R_1 and (b) R_2.

Fig.4 and Fig.5 show that PDGA outperforms SGA on both R_1 and R_2 with respect to both percentage of achieving optimum over 1000 runs against generations and mean percentage of optimal individuals in the population against generations over 1000 runs, especially during the early generations of GA's searching process. Fig.4 and Fig.5 also show that PDGA achieves more improvement over SGA on R_2 than on R_1.

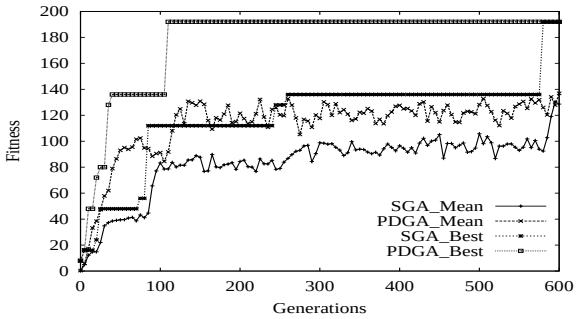
To further compare the dynamic searching process of PDGA and SGA, we give out the result of a typical run of PDGA and SGA on R_1 and R_2 in Fig.6 with respect to mean fitness and best fitness achieved against generations. The data are plotted every 5 generations. On R_1, PDGA achieves optimum at generation 85 with 9252 evaluations while SGA at generation 400 with 43264 evaluations. On R_2, PDGA achieves optimum at generation 108 with 11784 evaluations while SGA at generation 578 with 62360 evaluations. From Fig. 6 it can also be seen that PDGA outperforms SGA quite well especially during the early generations.

5 Conclusions

In this paper, based on the idea of diploid and dominance phenomenon widely existing in nature,



(a)



(b)

Fig.6 Mean and best fitness against generations for a typical run of PDGA vs. SGA on (a) R_1 and (b) R_2 .

we proposed a primal-dual genetic algorithm (PDGA) that operates on a pair of primal-dual chromosomes in the sense of Hamming distance. We compared the performance of PDGA over SGA based on the Royal Road functions, specially designed for testing the performance of GAs. The experiments show that PDGA outperforms SGA for different performance criteria, such as function evaluations to optimum, percentage of achieving optimum over a number of runs, and percentage of optimal members in the population for a run.

PDGA is proposed with the aim of improving GA's searching efficiency in the search space through primal-dual mapping. Here the mapping function has the key role in PDGA's performance. In this paper we take the Hamming distance as the primal-dual mapping function, which is a static mapping function. This mapping function works well during the early generations by shortening genetic operations performed on low fitness chromosomes and thus speed up GA's convergence. However, whence the mean fitness of the population becomes quite high, it loses its effect. For the future research on PDGA, we believe that dynamic mapping function that can adapt itself with GA's progressing will further improve PDGA's performance.

From the viewpoint of structure, PDGA shares the same framework with SGA, thus there can be many variations for PDGA as well as for SGA. Whatever variations (for example, the elitist scheme and the Sigma truncation scaling scheme used in this paper) that work well with SGA should also work well with PDGA. Thus combining so far developed advanced operators with PDGA is obviously one of the future research directions on PDGA.

Acknowledgements. This work was supported by the University of Leicester Research Fund 2001 under Grant FP15004, UK.

References:

[1] J. D. Bagley, *The Behaviour of Adaptive* Systems *Witch Employ Genetic and Correlation Algorithms*, Ph. D. Dissertation, University of Michigan, 1967.

[2] A. Brindle, *Genetic Algorithms for Function Optimization*, Doctoral Dissertation, University of Alberta, Edmonton, Canada, 1981.

[3] S. Forrest and M. Mitchell, Relative Building-Block Fitness and the Building-Block Hypothesis, *Foundations of Genetic Algorithms 2*, D. Whitley, Ed., 1993, pp. 109-126, Morgan Kaufmann.

[4] D. E. Goldberg and R. E. Smith, Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy, *Proc. of the 2nd International Conference on Genetic Algorithms and Their Applications*, J. J. Grefenstette, Ed., 1987, pp. 59-68, Lawrence Eribaum Associates.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan *Press*, 1975.

[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd edition, Springer-*Verlag*, New York, 1996.

[8] M. Mitchell, S. Forrest and J. H. Holland, The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance, *Proc. of the First European Conference on Artificial Life*, F. J. Varela and P. Bourgine, Eds., 1992, pp. 245-254, Cambridge, MA: MIT Press.

[9] M. Mitchell, *An Introduction to Genetic Algorithms*, Cambridge, MA: MIT Press, 1996.

[10] K. P. Ng and K. C. Wong, A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization, *Proc. 6th Int. Conf. on Genetic Algorithms*, L. J. Eshelman, Ed. 1995, pp. 159-166, Morgan Kaufmann.